The Journal of Systems and Software 110 (2015) 264–278

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Approximating closed fork-join queueing networks using product-form stochastic Petri-nets

Rasha Osman*, Peter G. Harrison

Department of Computing, Imperial College London, London SW7 2AZ, United Kingdom

ARTICLE INFO

Article history:
Received 11 May 2015
Revised 30 July 2015
Accepted 21 August 2015
Available online 2 September 2015

Keywords:
Product-form stochastic Petri nets
Reversed Compound Agent Theorem (RCAT)
NoSQL datastores

ABSTRACT

Computing paradigms have shifted towards highly parallel processing and massive replication of data. This entails the efficient distribution of requests and the synchronization of results provided to users. Guaranteeing SLAs requires the ability to evaluate the performance of such systems while taking the effect of non-parallel workloads into consideration. This can be achieved with performance models that are able to represent both parallel and sequential workloads. This paper presents a product-form stochastic Petri-net approximation of fork-join queueing networks with interfering requests. We derive the necessary conditions that guarantee the accuracy of the approximations and verify this through examples in comparison to simulation. We apply these approximate models to the performance evaluation of replication in NoSQL cloud datastores and illustrate the composition of large models from smaller models, thus facilitating the ability to model a range of deployment scenarios. We show the efficiency of our solution method, which finds the product-form solution of the models without the representation of the state-space of the underlying CTMC.

1. Introduction

The advent of heterogeneous cloud services and their complex underlying interdependencies necessitate the need for predictive, quantitative modelling of their performance as a prerequisite for their efficient design. As the size and interoperability of these systems increases, simulation rapidly becomes prohibitive necessitating the need for lightweight analytical models or their approximations to analyze and predict the performance and related costs of such emerging architectures. Cloud pay-for-use services, virtualization and high speed data generation and processing require dynamic, self-managing and autonomous systems that can effectively respond to changes in workloads by amending their configurations to meet SLAs at runtime. Analytical performance models and approximations are candidates for such implementations (Huebscher and McCann, 2008) within lightweight dynamic configuration monitors and tools.

The parallel processing paradigm has become the backbone of cloud and web services, in which requests are distributed to multiple processing nodes in parallel in order to decrease latency, in the case of high performance computing, and to increase availability in the context of web and cloud databases. This processing behavior is analogous to the fork-join structure in which arriving jobs are split to be processed at parallel nodes and then resynchronized after processing. Further, this fork-join split and synchronization behavior exhibits itself in

(1) the parallel processing of Big Data within the MapReduce framework in which intermediate results are combined to form the final result (Dean and Ghemawat, 2008);
(2) RAID disk storage (Patterson et al., 1988) that delivers high reliability coupled with high performance through the access of replicated data on multiple disks;
(3) the parallel invocation of composite web services within a service application (Menasce, 2004); and
(4) eventual consistency in NoSQL datastores in which copies of replicated data items are compared for consistency (Cattell, 2011; Stonebraker and Cattell, 2011).

Efficient utilization and energy efficiency require that infrastructure nodes are not dedicated to one application or service. Thus it is expected that in the above examples the infrastructure would process independent workloads that do not participate in the fork-join synchronizing process. These non-forking workloads will affect the performance, and thus the SLAs, of the forking requests. Such scenarios can be represented by fork-join systems *with interfering requests*; this allows for the performance analysis of realistic architectures and applications and does not isolate the forking requests from the effects of the environment.

In this paper, we present the approximate analytical solution of fork-join queueing systems with interfering requests utilizing the

* Corresponding author. Tel.: +44 (0) 20 7594 8331.
E-mail addresses: rosman@ieee.org, rosman@imperial.ac.uk (R. Osman), p.harrison@imperial.ac.uk (P.G. Harrison).

http://dx.doi.org/10.1016/j.jss.2015.08.036

product-form solution of stochastic Petri-nets (SPN) which we refer to as *RB-n-m replication blocks*. These models are applied to the performance analysis of NoSQL cloud datastores. Modelling replication in NoSQL datastores requires the representation of

(1) the processing of requests arriving at a node;
(2) the synchronization of requests for multiple copies of a data item and;
(3) the routing of requests based on the number of copies stored across the cluster.

NoSQL datastores cater mainly to distributed Web 2.0 applications (Stonebraker and Cattell, 2011) which require low latency with a weaker guarantee of data consistency (Cattell, 2011). Thus datastore performance depends on the ability of the nodes to cooperate in satisfying user requests in a timely fashion.

The primary technique used to solve the approximate fork-join queueing models is the Reversed Compound Agent Theorem (RCAT) (Harrison, 2003), which provides a unified framework for the derivation of product-form solutions for cooperating Markovian processes by representing the cooperation as RCAT *rate equations*. We approximate the fork-join queueing system as a certain type of synchronizing product-form stochastic Petri nets called *building blocks* (Balsamo et al., 2012) and derive the assumptions needed to accurately represent the performance measures of the fork-join queueing system. Automated direct solutions of the RCAT rate equations have not been attempted, as it has been deemed difficult due to the non-linearity of the equations (Balsamo et al., 2010a). In this paper, we formulate the solution of the RCAT rate equations into a non-linear optimization problem and illustrate the scalability and flexibility of the technique in comparison to current methods in the literature.

The contributions of this paper are the following:

- We present an approximation of fork-join queueing networks with interfering requests based on product-form SPNs, which we have called RB-n-m replication blocks. We give the conditions which guarantee the accuracy of these approximations and give counter examples when they exist. We validate the solution of the RB-n-m replication blocks in comparison to simulation for any n places and any m partial or full forking requests.
- We apply the product-form RB-n-m replication blocks to replication in NoSQL datastores and analyze the performance of these systems. In addition, we show how to compose larger models from smaller RB-n-m models to evaluate non-centralized scenarios. At present, performance modelling has concentrated on traditional relational databases (Osman and Knottenbelt, 2012), using mainly queueing networks, (e.g. Nicola and Jarke (2000); Elnikety et al. (2009); Di Sanzo et al. (2010); Osman et al. (2011)) and queueing Petri nets, e.g. (Osman et al., 2013; Coulden et al., 2013). Recent work (Osman and Piazzolla, 2014; Gandini et al., 2014) in modelling NoSQL replication uses simulation based techniques to solve the models.
- We have formulated the RCAT rate equations of our models as a non-linear optimization problem, which finds the product-form solution of the models without the representation of the state-space of the underlying CTMC. We show that our method is more flexible and scalable in comparison to existing RCAT automated solution methods in the literature.

The rest of this paper is organized as follows. Section 2 presents the background and related work for fork-join systems. In Section 3, we present the details of the product-form approximation and conditions for accuracy. We illustrate the generatation of the RCAT rate equations in Section 4. Validation of the models is presented in Section 5. We model and evaluate the performance of NoSQL datastores in Section 6. Finally, we conclude the paper and provide directions for future work in Section 7.

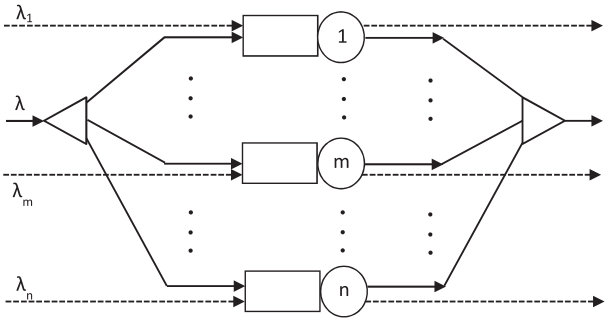

Fig. 1. An n-queue fork-join queueing system with full fork and interfering requests.

2. Background and related work

2.1. Fork and join systems

Fig. 1 details an n-branch fork-join queueing system where each branch is represented by an $M/M/1$ queue. The system has two types of arrivals: *forking* and *interfering*. Forking requests arrive at the forking point and are split into m ($m > 1$) independent tasks to be served in parallel at all of the m queues. Tasks that have completed their service wait at the join point until all m jobs have competed their service and then re-join and leave the system. Interfering jobs are independent arrivals to any of the n queues that do not fork and thus affect the processing of forked tasks as shown in Fig. 1. In this paper we refer to forked arrivals as replicated requests and interfering arrivals as non-replicating requests.

Arriving jobs can be forked to $m < n$ queues, known as partial fork, or $m = n$ queues known as full-fork. Fig. 1 shows an example of full-forked tasks. Partial forked jobs will randomly choose m of the n queues for service. Dynamic forking is when a forked request will arrive with a different probability to each of the m queues. Fork-join systems can be homogenous, i.e., all service times for forked and interfering requests are the same or heterogeneous in which their service times differ at each queue. In this paper, we examine homogeneous systems and partially heterogeneous systems in which the service times for the forking (interfering) requests are the same at each queueing center. We present cases for full fork and partial fork, both with interfering requests.

2.2. Related work

The synchronized arrivals and processing of forked requests make the exact analysis of fork-join queueing systems intractable, as the state space of the model becomes impractical to solve. Flatto and Hahn (1984) provide an exact solution for an open two-way markovian fork-join queue with heterogeneous servers by solving the underlying CTMC. This was extended by Baccelli and Makowski (1985), Baccelli (1985) to derive the exact solution for an M/G/2 fork-join system. The performance measures of fork-join queueing systems with more than two service centers are derived using approximation techniques. Most of these studies derived bounds on the performance for single-class full-fork systems with homogeneous service centers. Baccelli and Makowski (1989), Varma and Makowski (1994) approximate the bounds on the mean response times for full fork open systems. Varki (2001), Varki et al. (2013) calculate response time bounds for open and closed fork-join systems and extend Mean-Value Analysis (MVA) for such closed systems in Varki (1999).

Maximum order statistics are used to approximate the synchronization time of heterogeneous full fork systems in Lebrecht and Knottenbelt (2007); Harrison and Zertal (2003). Chen (2001) applies a bubble sort analysis in combination with simulation to approximate an upper bound for the mean response time of an open full-fork

homogeneous queueing system. A matrix geometric solution is applied to derive the upper and lower bounds of the performance measures of a heterogeneous full-fork open system in Balsamo et al. (1998) and to calculate the moments of the mean queue lengths in Balsamo and Mura (1997). Casale et al. (2008) apply a noniterative analysis technique to approximate the performance measures for a closed full-fork queueing system.

Few studies investigate the performance of dynamic fork-join systems. For example, Varki et al. (2012) calculate the pessimistic and optimistic bounds and approximation for the mean response time of a fork-join queue with dynamic forking. Kumar and Shorey (1993) calculate bounds on the mean response times for a fork-join system in which the number of forked jobs is random. Both these works assume a single class homogeneous system. The work in Alomari and Menasce (2014) provides an MVA approximation for dynamic and full fork joins for a multi-class homogenous and heterogeneous open and closed fork-join queues. The customers in the approximation are distinguished by their forking behaviour and service times.

2.2.1. Fork-join systems with interfering requests

Nelson and Tantawi (1988) build on the work in Flatto and Hahn (1984) to derive an approximate closed-form expression for the mean response time for a n-way fork-join system, $2 \leq n \leq 32$, with homogenous servers. Interfering read requests are assumed to arrive uniformly to the n servers and write requests are processed as a full-fork to all n servers. The work in Thomasian and Menon (1994); 1997); Thomasian (1997) present an approximation of mean response times for fork-join requests in RAID5 disk arrays in degraded mode in which interfering requests arrive at the $n-1$ surviving disks in addition to a $(n-1)$-full-fork request that is used to reconstruct the n^{th} failed disk.

The work presented in this paper approximates heterogeneous closed fork-join queueing systems with interfering requests by deriving the mean performance measures for full and partial fork scenarios. We approximate the n-node fork-join system by a closed n-place product-form SPN. The models are represented by a set of non-linear equations (RCAT rate equations) that describe the system. Our method is less computationally intensive in comparison to the iterative and non-iterative methods present in the literature, as the structure of the non-linear equations allows for efficient solutions using available solvers. In the rest of this paper we will refer to a fork-join queueing system with interfering requests simply as a fork-join queueing system.

2.3. Product-form SPNs

The Reversed Compound Agent Theorem (RCAT) (Harrison, 2003) is a generic framework to establish the existence and derivation of product-form solutions. RCAT specifies a set of sufficient conditions on the specification of the cooperating Markovian processes, if applicable, ensures a product-form solution exists. The *synchronization* between the cooperating components is represented by a system of equations referred to as the *rate equations* of the model which are formulated from the *reversed rates* of the model. If the system of rate equations has a solution then the model is in product-form. A general introduction to RCAT is provided in the Appendix. In this paper, we derive and solve the RCAT rate equations for product-form closed SPNs that approximate the performance measures of the corresponding closed fork-join queueing networks.

RCAT refers to the structure used to analyse SPNs in product-form as *building blocks* (BBs) (Balsamo et al., 2012). A BB-N consists of a set of places $P_1, \ldots, P_N$, a set $\mathcal{T}_I$ of input transitions whose input vectors are null and a set $\mathcal{T}_O$ of output transitions whose output vectors are null. All the arcs have multiplicity 1.

Definition 1. (Building Block (BB) (Balsamo et al., 2012)) Given an ordinary (connected) SPN $\mathcal{S}$ with a set of transitions $\mathcal{T}$ and set of N places $\mathcal{P}$. Define $\mathbf{I}(T)$ as the input vector and $\mathbf{O}(T)$ as the output vector of transition T. Then $\mathcal{S}$ is a *building block* if it satisfies the following conditions:

1. *All transitions are either input or output transitions*: For all $T \in \mathcal{T}$ then either $\mathbf{O}(T) = \mathbf{0}$ or $\mathbf{I}(T) = \mathbf{0}$. In the former case, we say that $T \in \mathcal{T}_O$ is an *output transition* while in the latter we say that $T \in \mathcal{T}_I$ is an *input transition*. Note that $\mathcal{T} = \mathcal{T}_I \cup \mathcal{T}_O$ and $\mathcal{T}_I \cap \mathcal{T}_O = \phi$, where $\mathcal{T}_I$ is the set of input transitions and $\mathcal{T}_O$ is the set of output transitions.
2. For each $T \in \mathcal{T}_I$, there exists $T' \in \mathcal{T}_O$ such that $\mathbf{O}(T) = \mathbf{I}(T')$ and vice versa.
3. *The SPN must be connected*: Given two places $P_i, P_j \in \mathcal{P}, 1 \leq i, j \leq N$, there exists a transition $T \in \mathcal{T}$ such that the component i and j of $\mathbf{I}(T)$ or of $\mathbf{O}(T)$ are non-zero.

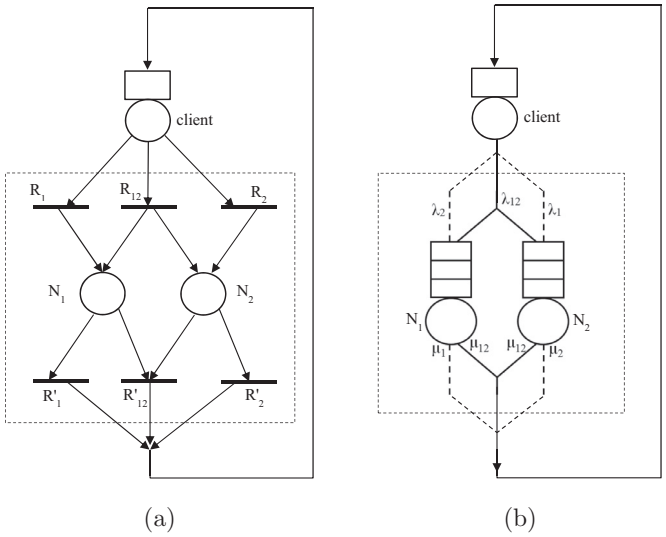

Fig. 2. The (a) RB-2-2 replication block and the corresponding (b) fork-join queueing system. The area within the dotted rectangle in (a) represents the BB-2 SPN.

Figs. 2 and 3 give examples of models containing BBs consisting of two (BB-2) and three (BB-3) places. In this paper, we use N_y to refer to a place with index y and $R_y(R'_y)$ to denote an input (output) transition, where y is the set of place-indices that the input (output) transition produces (consumes). The arrival rate at place N_y is λ_y and the service rate is μ_y. The sufficient conditions for the product-form of a BB is given in Theorem 1 (Balsamo et al., 2012) and the throughput (reversed rate) of the output transition in Lemma 1 (Balsamo et al., 2012).

Theorem 1. *Consider a BB with N places. Let* $\rho_y = \lambda_y/\mu_y$ *for* $R_y, R'_y \in \mathcal{T}$, $|y| \geq 1$. *If the following system of equations has a unique solution* ρ_i, $(1 \leq i \leq N)$:

$$\begin{cases} \rho_y = \prod_{i \in y} \rho_i & \forall y: \quad R_y, R'_y \in \mathcal{T} \wedge |y| > 1 \\ \rho_i = \dfrac{\lambda_i}{\mu_i} & \forall i: \quad R_i, R'_i \in \mathcal{T}, 1 \leq i \leq N \end{cases} \tag{1}$$

then the net's balance equations - and hence stationary probabilities when they exist - have a product-form solution:

$$\pi(m_1, \ldots, m_N) \propto \prod_{i=1}^{N} \rho_i^{m_i}. \tag{2}$$

Lemma 1. *In a product-form BB that satisfies the conditions of Theorem 1, the throughput (reversed rate) of every output transition* $R'_y \in \mathcal{T}_O$ *is* λ_y, *i.e. the rate of the corresponding input transition.*

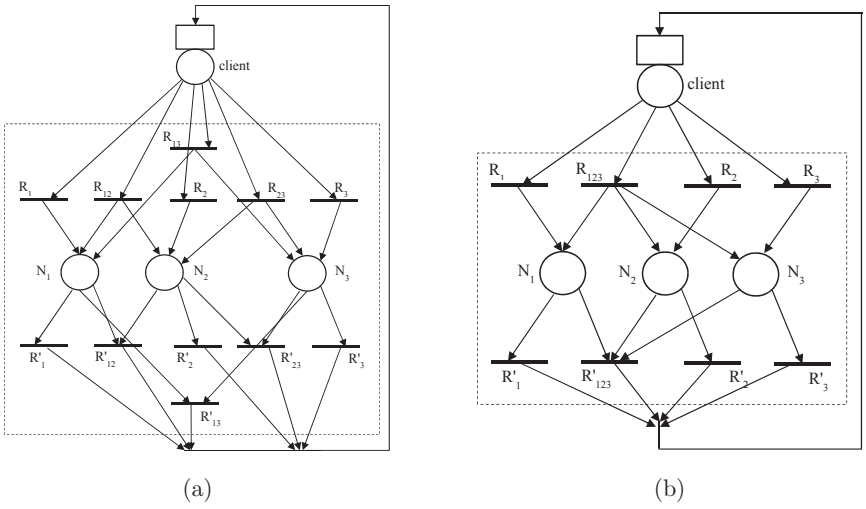

Fig. 3. The (a) RB-3-2 and (b) RB-3-3 replication blocks. The area within the dotted rectangle represents the BB-3 SPNs.

For example, the BB-2 in Fig. 2(a) will satisfy the conditions of Theorem 1 when

$$\rho_{12} = \rho_1 \rho_2.$$

Further, we refer to the reversed rate of R'_y as x_y and thus the BB-2 in Fig. 2(a) will be in product-form when

$$\frac{x_{12}}{\mu_{12}} = \frac{x_1}{\mu_1}\frac{x_2}{\mu_2}.$$

The rest of this paper will use this form to express the product-form conditions of a BB-N in order to avoid confusion with the stability conditions of the models presented in the following sections.

3. Approximation of fork-join systems

Fig. 2(b) shows an example of a two-node closed fork-join queueing network. In this example, the client generates requests to the fork-join queues, in which a request is either sent to queue N_1 with rate λ_1, queue N_2 with rate λ_2, or forked to both queues with rate λ_{12}. These arrival rates depend on the routing probabilities specified for the output of the client-node. We assume that a forked request is processed with service rate μ_{12} at both queues and a non-forked request is processed with rate $\mu_1 = \mu_2$ at each individual queue. These assumptions produce two customer classes from the perspective of the fork-join queues (forked and non-forked) and one customer class from the perspective of the client and the complete system. When $\mu_1 = \mu_2 = \mu_{12}$, the system will reduce to a single customer class in relation to processing. These assumptions are comparable to actual systems, e.g., replicated databases or RAID clusters, in which the request processing time can be the same but the behavior of the request in the system differs, based on whether the data is allocated to one or more servers/disks or the processing times for forked requests differ from the corresponding non-forked requests due to synchronization. In this section, we illustrate how to approximate such a closed n-node fork-join queueing system using product-form SPNs constructed from BBs.

The BB-2 of Fig. 2(a) represents the approximation of the equivalent two-node fork-join system of Fig. 2(b). For the BB-2, the fork process is represented by the input transition R_{12} and the join process is represented by the output transition R'_{12}. Evidently, this does not faithfully model the join operation, e.g., when R_{12} deposits forked requests into N_1 and N_2, all output transitions become enabled and thus one transition is chosen randomly to fire. If R'_1 or R'_2 fires, this would result in an output of a non-forked request instead of the intended forked request, thus generating one additional request in the system. Conversely, two single non-forked arrivals from R_1 and R_2 may be removed from N_1 and N_2 together by a firing of R'_{12}, decreasing the population by one. In the case of a product-form BB-2, the mean token population stays the same when the fork-join transitions are added, but the throughput obviously increases to the sum of all the input transition rates – both forking and non-forking. Indeed, we observe in simulation tests that the closed network of Fig. 2(a) has a much higher system throughput than that of the corresponding fork-join system. For a BB-n to accurately approximate the corresponding n-node fork-join queueing system, the effect of the non-synchronizing transitions, R'_1 and R'_2 in the case of the BB-2, needs to be minimized, such that the resulting performance measures of the BB-n in the steady state are sufficiently close to those of the n-node fork-join queueing system. In the following, we derive the required constraints. The term synchronizing input (output) transition will be used interchangeably to refer to the forking (joining) transitions in the SPNs.

3.1. Definitions

We approximate the quantitative behavior of multi-class fork-join (replicated) systems using product-form SPNs. A fork-join queueing system is represented as a closed model with an IS client node representing incoming requests and a BB-n representing the nodes (queues) of the fork-join system, as in Fig. 2(a). We refer to these closed models as *replication blocks* (RBs) and use the notation *RB-n-m* to refer to a replication block that represents an n-node fork-join queueing system with fork policy m, where $m = 2, \ldots, n$, i.e., all possible partial forks and a full fork for the forking customer class. For consistency, we refer to the fork policy as the replication factor. In addition, all n nodes in the replication block receive non-replicating (i.e., non-forking) requests. In this work, we consider systems in which the service rates of the replicated (non-replicated) requests are identical.

To simplify the calculation of the performance measures of the RB-n-m and avoid the calculation of the normalization constant, we follow the convention in Lladó and Harrison (2011), Balsamo et al. (2012) by considering each place N_i in the BB-n as an independent $M/M/1$ queue. This approximation is exact when no forking customers arrive into the system. For example, in the RB-2-2 in Fig. 2(a), if the synchronizing input and output transitions (R_{12}&R'_{12}) are suspended and accordingly the forking class in Fig. 2(b), then the two models will be equivalent, i.e., two independent $M/M/1$ queues with one customer class. However, when the synchronizing input and output transitions are reactivated, the fork-join queueing system will receive two customer classes that will be processed independently, while the SPN

will receive one token type that will be processed randomly, irrespective of whether it was the result of firing a synchronizing or non-synchronizing input transition. For this approximation to be equivalent to the fork-join system we must consider the effect of the lack of distinction between the forking and non-forking customers in the RB-n-m.

First, the stability condition $\rho_i < 1$ for each place N_i should consider the arrivals and departures of both the synchronizing and non-synchronizing customers similar to the fork-join system. This will prevent the RB-n-m places from becoming saturated. To achieve this, we distinguish between the arrival and departure rates of synchronizing and non-synchronizing tokens. We present the following well-known stability condition for each place N_i of the SPN within the RB-n-m as a definition:

Definition 2. The stability condition for the product-form BB-n defined in Theorem 1 when approximating each place as an $M/M/1$ queue is:

$$\rho_i = \frac{x_i}{\mu_i} + \sum_{j \neq i} \frac{x_j}{\mu_j} < 1$$

Here i is the index of the non-synchronizing input transition of node N_i and j is the index of the synchronizing input transitions to place N_i where j represents the set of place indices that receive tokens from the synchronizing transition including N_i. Definition 2 represents the value of ρ_i used to calculate the performance measures of the RB-n-m in Section 4.3. As an example, consider the BB-2 in Fig. 2(a), in which the stability condition for the places will be:

$$\rho_i = \frac{x_i}{\mu_i} + \frac{x_{12}}{\mu_{12}} < 1, \text{ where } i = 1, 2.$$

Second, we must limit the firing of the non-synchronizing output transitions in order to minimize the rate of production of extra tokens in the system. This is accomplished by placing a constraint on the reversed rates of the non-synchronizing output transitions such that, in the steady state, the RB-n-m is equivalent to the fork-join queueing system. We would expect the solution of the RB-n-m models to be an accurate approximation when $\mu_i < \mu_j$. From experimentation and simulations, we have estimated an upper bound for the reversed rates of the non-synchronizing output transitions R'_i, such that, in the steady state, the effect of the non-synchronizing transitions is minimized. This bound is formally defined as:

Conjecture 1. *A product-form BB-n with stability condition as defined in Definition 2, will make the effect of the non-synchronizing transitions negligible, in the steady state, thus approaching the behavior of an n-node fork-join queueing system, when:*

$$\frac{x_i}{\mu_i} < \frac{1}{n_i}, \text{ where } i = 1, \ldots, N,$$

and n_i is the total number of input transitions entering place N_i.

The combination of the previous constraints and the values of the output transition rates determine the value of the reverse rates, which in turn, determine the mean number of tokens/customers in each place in the steady state. When the mean number of customers in the system is large, we would expect the RB-n-m to be an accurate approximation as, in equilibrium, any inaccuracies will tend to average out. Thus, a lower bound on the mean number of customers in the system should be defined, such that, in equilibrium, the performance of the SPN and the fork-join system are identical. From experimentation, we have found that when the mean number of customers is greater than the number of nodes in the system, the RB-n-m will be an excellent approximation of the corresponding n-node fork-join queueing system. We formally express the relationship between the output transition rates, the previous constraints (1 & 2) and the lower bound on the mean number of customers in the system as follows.

Conjecture 2. *For an RB-n-m with an N place BB-n and stability condition of Definition 2, define $\mu_i(\mu_j)$ to be the service rate of the non-synchronizing (synchronizing) output transitions for place N_i and n_i the total number of input transitions to place $N_i, (1 \leq i \leq N)$. The RB-n-m is an accurate approximation of the corresponding n-node closed fork-join queueing system with mean number of customers C in the system, iff:*

$$\frac{x_i}{\mu_i} < \frac{1}{n_i},$$
$$\mu_i \neq \mu_j, \text{ and}$$
$$C > N.$$

When $\mu_i = \mu_j$, then $C \gg N$.

In Sections 5.1 & 5.2, we give examples and counter examples for the above conjecture and show its applicability when modelling replication in NoSQL datastores in Section 6. In the next Section, we derive the RCAT rate equations for two and three node replication blocks for different fork policies, then generalize to n-node systems.

4. Modelling fork-join constructs

4.1. The RB-2-2 replication block

Fig. 2(a) shows a replication block with two places/nodes (N_1 and N_2) that participate in a synchronization. The RB-2-2 represents a two node cluster, in which the replication factor is two. The client is represented by an IS queue. A client sends two classes of requests: *non-replicated* requests, which are randomly routed to each node with probability p_i, where $i = 1, 2$, and *replicated* requests, which are routed to both nodes in parallel with probability p_{12}. Arrivals of non-replicated requests are represented by the non-synchronizing timed-transitions R_1 and R_2. Arrivals of replicated requests are represented by the synchronizing transition R_{12}, in which the indices represent the synchronizing nodes. All nodes in the cluster are identical and thus processing times are the same: non-replicated request processing is represented by the output transitions R'_1 & R'_2 and replicated request processing is represented by the output transition R'_{12}. After processing, all requests are routed back to the client.

Using RCAT terminology, in the RB-2-2 the active actions are the service completions at the client queue or the firing of an output transition at the replication block. The reversed rates are the combined arrival rates at the client queue or the input transitions to the replication block. Assume the service rates at the client queue and the output transitions are μ_T, μ_1, μ_2, μ_{12} and the arrival rates (reversed rates) at the client queue and the input transitions are x_T, x_1, x_2, x_{12}, respectively (direct application of Lemma 1). Then the rate equations for the RB-2-2 are as follows.

$$x_1 = p_1 x_T,$$
$$x_2 = p_2 x_T,$$
$$x_{12} = p_{12} x_T,$$
$$x_T = x_1 + x_2 + x_{12}.$$

From Theorem 1 the RB-2-2 is in product-form when satisfying:

$$x_{12} = \mu_{12} \frac{x_1}{\mu_1} \frac{x_2}{\mu_2}.$$

Under the condition $p_1 + p_2 + p_{12} = 1$. To approximate a two-node fork-join queue, the rate equations are solved with the additional condition (Conjecture 1):

$$\frac{x_i}{\mu_i} < \frac{1}{2}, \text{ where } i = 1, 2.$$

To derive the performance measures of the RB-2-2, we calculate the stability condition taking into account the rate of non-replicated and

replicated requests arriving and departing from each place. Therefore, the stability condition for each place is (Def. 2):

$$\rho_i = \frac{x_i}{\mu_i} + \frac{x_{ik}}{\mu_{ik}} < 1, \text{ where } i, k = 1, 2 \text{ and } k \neq i.$$

and the stability condition for the IS client is:

$$\rho_i = \frac{x_i}{\mu_i} < 1, \text{ where } i = T.$$

4.2. The RB-3-m Replication Block

Fig. 3(a & b) shows the RB-3-2 & RB-3-3 replication blocks. Here the cluster size is 3 with the places N_1, N_2 & N_3 representing the nodes. The data is replicated two times in RB-3-2 (Fig. 3(a)) and three times in RB-3-3 (Fig. 3(b)). The non-replicated request arrivals are represented by transitions R_1, R_2 & R_3 with arrival probabilities $p_i, i = 1, 2, 3$. Processing is represented by the output transitions R'_1, R'_2 & R'_3 with rates $\mu_i, i = 1, 2, 3$. For the RB-3-2, the replicated request arrivals are represented by the $\binom{3}{2}$ number of synchronizing transitions labeled R_{ik} with arrival probability of p_{ik} and corresponding output transitions R'_{ik} where $k = 1, 2, 3$ and $k \neq i$. For the RB-3-3, the replicated request arrivals are represented by the $\binom{3}{3}$ transition, in this case R_{123} with arrival probability p_{123} and corresponding output transition R'_{123}. For both replication blocks the combined service rates are routed back to the client.

Let A be the set of node indices of RB-3-2 and C be the set of all 2-combinations of A. Hence, the rate equations for the RB-3-2 are:

$$x_i = p_i x_T, i \in A,$$
$$x_j = p_j x_T, j \subset C,$$
$$x_T = \sum_{i \in A} x_i + \sum_{j \subset C} x_j.$$

The RB-3-2 is in product-form when satisfying:

$$x_j = \mu_j \frac{x_k}{\mu_k} \frac{x_l}{\mu_l}, j \subset C \,\&\, (k, l) \in j.$$

Under the condition:

$$\sum_{i \in A} p_i + \sum_{j \subset C} p_j = 1,$$

and the condition for approximating a three-node fork-join system:

$$\frac{x_i}{\mu_i} < \frac{1}{n_i}, \text{ where } i \in A, n_i = |j| + 1, j \subset C, i \in j.$$

In this case, $n_i = 3$, representing two replicating and one non-replicating input transitions to each place. The stability conditions for the RB-3-2 are:

$$\rho_i = \frac{x_i}{\mu_i} + \sum_{\substack{j \subset C \\ i \in j}} \frac{x_j}{\mu_j} < 1, i \in A,$$

$$\rho_i = \frac{x_i}{\mu_i} < 1, i = T.$$

The RB-3-2 has seven rate equations and three conditions for product-form. Redefining A as the set of node indices of RB-3-3 and C as the set of all 3-combinations of A. The rate equations for the RB-3-3 will be the same as those for the RB-3-2, except producing five rate equations and one condition for product-form.

4.3. The general RB-n-m replication block

In general, a replication block RB-n-m has a SPN net with n places and $n + \binom{n}{m}$ transitions. N represents the cluster size and m is the replication factor and $m \leq n$. Let A be the set of node indices of the RB-n-m and C be the set of all m-combinations of A. Requests arrive from an IS client with rate μ_T. The n non-replicated requests arrive with probability p_i to the input transition $R_i, i \in A$. The $\binom{n}{m}$ replicated requests arrive with probabilities p_j to the input transitions R_j where $j \subset C$. The corresponding output transitions for non-replicated and replicated requests are R'_i & R'_j with service rates μ_i & μ_j. The output service rates combine to form the arrival rate at the IS client. The complete net will have $n + 1 + \binom{n}{m}$ rate equations as follows:

$$x_i = p_i x_T, i \in A, \tag{3}$$

$$x_j = p_j x_T, j \subset C, \tag{4}$$

$$x_T = \sum_{i \in A} x_i + \sum_{j \subset C} x_j. \tag{5}$$

The number of conditions for product-form for an RB-n-m replication block is $\binom{n}{m}$ given by:

$$x_j = \mu_j \prod_{k \in j} \frac{x_k}{\mu_k}, j \subset C. \tag{6}$$

Under the condition:

$$\sum_{i \in A} p_i + \sum_{j \subset C} p_j = 1, \tag{7}$$

and the condition for approximating an n-node fork-join system:

$$\frac{x_i}{\mu_i} < \frac{1}{n_i}, \text{ where } i \in A, n_i = |j| + 1, j \subset C, i \in j. \tag{8}$$

The stability conditions for the RB-n-m are:

$$\rho_i = \frac{x_i}{\mu_i} + \sum_{\substack{j \subset C \\ i \in j}} \frac{x_j}{\mu_j} < 1, i \in A, \tag{9}$$

$$\rho_i = \frac{x_i}{\mu_i} < 1, i = T. \tag{10}$$

4.3.1. Performance measures

From Eqs. (3)–(6), the accurracy condition (8) and the stability conditions (9 & 10) the unconditional product-form in equilibrium for the RB-n-m is given by:

$$\pi(m_1, \ldots, m_n, m_T) = \frac{\rho_T^{m_T} e^{-\rho_{m_T}}}{m_T!} G \prod_{i \in A} \rho_i^{m_i} \tag{11}$$

G is the normalizing constant, which is calculated as the product of the normalizing constants for each place $N_i, i \in A$, as we consider each place in the BB-n-m as an $M/M/1$ queue:

$$G = \prod_{i \in A} (1 - \rho_i)$$

Renaming m_T to m_{n+1}, Eq. (11) becomes:

$$\pi(m_1, \ldots, m_n, m_{n+1}) = \frac{e^{-\rho_{m_{n+1}}}}{m_{n+1}!} \prod_{i=1}^{n} (1 - \rho_i) \prod_{i=1}^{n+1} \rho_i^{m_i} \tag{12}$$

From Eq. (12), the mean queue length for each place in the net is:

$$q_i = \frac{\rho_i}{(1 - \rho_i)}, i \in A, \; q_{n+1} = \rho_{n+1}. \tag{13}$$

The throughput of the complete system is:

$$Thpt = \rho_{n+1} \mu_{n+1} = x_{n+1}. \tag{14}$$

From Little's Law, the response time for the closed system is:

$$RT = \frac{\sum_{i=1}^{n+1} q_i}{Thpt} - \frac{1}{\mu_{n+1}}. \tag{15}$$

In the next sections, we use these results to compare the analytical solution of the RB-n-m replication blocks to the simulation of the corresponding n-node fork-join queueing systems.

Given:
- n: the number of nodes
- m: the replication factor
- A: the set of node indices
- C: the set of all m-combinations of A
- $\mu_i, i \in A$: the service rates of the non-synchronizing output transitions
- $\mu_j, j \subset C$: the service rates of the synchronizing output transitions
- μ_{n+1}: the client think rate
- u_i: the maximum utilization, $i = 1, \ldots, n+1, u_i < 1$

Variables:
- x_i: the reversed rates, $i \in A \cup \{n+1\}$ or $i \subset C, x_i > 0$
- p_i: the probability of request arrivals from the client, $i \in A$ or $i \subset C, p_i > 0, \sum p_i = 1$

Maximize:
- x_{n+1}: the throughput of the model

Subject to: The rate equations:

$$x_i = p_i x_{n+1}, i \in A$$

$$x_j = p_j x_{n+1}, j \subset C$$

The product-form conditions:

$$x_j = \mu_j \prod_{k \in j} \frac{x_k}{\mu_k}, j \subset C$$

The accuracy conditions:

$$\frac{x_i}{\mu_i} < \frac{1}{n_i}, \text{ where } i \in A, n_i = |j| + 1, j \subset C, i \in j$$

Fig. 4. The non-linear optimization model for the RB-n-m replication block.

5. Solving the RB-n-m replication block

Assuming the service times are known, the RB-n-m requires $\binom{n}{m} + n$ inputs to be chosen such that the conditions of product-form are satisfied. There are n independent inputs which represent the non-replicating transitions. Hence, we can independently choose the n independent input probabilities for the non-replicating transitions, after which the remaining $\binom{n}{m}$ probabilities can be chosen to guarantee product-forms for the RB-n-m. For an RB-2-2 this is straightforward and the rate equations can be solved. However, as the number of places increases the process of choosing the independent n probabilities that will satisfy the product-form conditions and produce a solution for the rate equations becomes tedious.

The rate equations, the product-form conditions and the accuracy condition for the RB-n-m form a system of non-linear equations. To find a general solution to this non-linear system we formulate it as a non-linear optimization problem. The objective is to find the $\binom{n}{m} + n$ probabilities (Eq. (7)) and the $n + 1 + \binom{n}{m}$ values of the x_is that maximize the throughput (Eq. (14)) of the net. We have chosen to maximize the throughput as it is analogous to maximizing the reversed (arrival) rates of the nodes, as evident from Eq. (5). This is subject to the constraints of: (a) the $n + 1 + \binom{n}{m}$ rate equations (Eqs. (3)–(5)), (b) the $\binom{n}{m}$ product-form conditions (Eq. (6)), (c) the n accuracy conditions (Eq. (8)) and (d) the $n + 1$ stability conditions (Eqs. (9) & (10)). The input to the optimization problem is the number of nodes, n, the replication factor, m, the service demands of the output transitions, μ_i, the client think rate, μ_T and the maximum utilization, $u_i, i = 1, \ldots, n+1$. A specification of the non-linear optimization model is in Fig. 4.

5.1. Validation

To solve the non-linear optimization problem, we parameterized the model by specifying the service rate for all the non-replicating output transitions with $\mu_i = 5$ and $\mu_j = 12$ for the replicating output transitions. The client think rate was $\mu_T = 0.5$ and the utilization was set to $u_i \leq 0.99$ ($i \in A$ or $i = T$). Solving the non-linear system, we obtained the values for the probabilities and the reverse rates that satisfy the rate equations and the product-form and accuracy conditions. Table 1 details the computed values for the RB-2-2, RB-3-m and RB-4-m, where $m = 2, \ldots, n$, replication blocks. We observe that the routing probabilities for the R_is are equal and thus their reversed rates are equal. This also applies to the R_js. This result is independent of the service rates, as it holds even when $\mu_i = \mu_j$.

From the results in Table 1, we calculated the performance measures for each model. To validate the solution of the RB-n-m blocks we compared the results provided by the solver to a discrete event simulation of the corresponding n-node fork-join queueing system. The simulation was parameterized with the service rates for each RB-n-m and the probabilities and mean number of customers acquired from the solution of the non-linear system. The performance measures calculated for the RB-2-2, RB-3-m & RB-4-m nets, where $m = 2, \ldots, n$, were compared to the simulation of the corresponding n-node fork-join queueing system at 95% confidence interval. Table 2 confirms the excellent agreement between the product-form solution in comparison to the simulation, with the RB-n-m accurately approximating the performance of the corresponding fork-join systems, giving errors of less than 5% for the mean queue length at each place, the overall mean throughput of the system and the utilization

Table 1
Values for the reversed rates and probabilities for the RB-2-2, RB-3-m, RB-4-m replication blocks with $\mu_i = 5$ for the non-replicating output transitions, $\mu_j = 12$ for the replicating output transitions and $\mu_T = 0.5$ for the client think rate.

RB-2-2	index	x	prob	RB-3-2	index	x	prob	RB-3-3	index	x	prob
	1	2.4998	0.3125		1	1.6665	0.1852		1	2.4998	0.2778
	2	2.4998	0.3125		2	1.6665	0.1852		2	2.4998	0.2778
	3	2.9994	0.3750		3	1.6665	0.1852		3	2.4998	0.2778
	T	7.9989			4	1.3331	0.1481		4	1.4996	0.1666
					5	1.3331	0.1481		T	8.9988	
					6	1.3331	0.1481				
					T	8.9987					
RB-4-2	index	x	prob	RB-4-3	index	x	prob	RB-4-4	index	x	prob
	1	1.2499	0.1316		1	1.2499	0.2174		1	2.4998	0.2326
	2	1.2499	0.1316		2	1.2499	0.2174		2	2.4998	0.2326
	3	1.2499	0.1316		3	1.2499	0.2174		3	2.4998	0.2326
	4	1.2499	0.1316		4	1.2499	0.2174		4	2.4998	0.2326
	5	0.7499	0.0789		5	0.1874	0.0326		5	0.7497	0.0697
	6	0.7499	0.0789		6	0.1874	0.0326		T	10.7487	
	7	0.7499	0.0789		7	0.1874	0.0326				
	8	0.7499	0.0789		8	0.1874	0.0326				
	9	0.7499	0.0789		T	5.7493					
	10	0.7499	0.0789								
	T	9.4986									

Table 2
Comparison between the analytical solution of the RB-2-2, RB-3-m, RB-4-m replication blocks and simulation of the corresponding n-node fork-join queueing systems.

		Mean queue length					Mean queue length		
RB-2-2	Place	Model	Sim	Error %	RB-4-2	Place	Model	Sim	Error %
	Client	15.9978	16.7330	−4.39		Client	18.9972	19.1440	−0.77
	N1	2.9984	2.9660	1.09		N1	0.7776	0.8060	−3.53
	N2	2.9984	2.9670	1.06		N2	0.7776	0.8060	−3.53
						N3	0.7776	0.8050	−3.41
	Throughput	7.9989	8.3680	−4.41		N4	0.7776	0.8070	−3.65
	Response time	0.7497	0.6290	19.19					
	Utilization	0.7499	0.7840	−4.35		Throughput	9.4986	9.5720	−0.77
						Response time	0.3275	0.2980	9.88
RB-3-2	Place	Model	Sim	Error %		Utilization	0.4374	0.4410	−0.81
	Client	17.9974	18.5060	−2.75					
	N1	1.2496	1.3130	−4.83	RB-4-3	Place	Model	Sim	Error %
	N2	1.2496	1.3150	−4.97		Client	11.4985	11.4570	0.36
	N3	1.2496	1.3170	−5.12		N1	0.4221	0.4140	1.97
						N2	0.4221	0.4150	1.72
	Throughput	8.9987	9.2520	−2.74		N3	0.4221	0.4150	1.72
	Response time	0.4166	0.3780	10.21		N4	0.4221	0.4150	1.72
	Utilization	0.5555	0.5710	−2.72					
						Throughput	5.7493	5.7280	0.37
RB-3-3	Place	Model	Sim	Error %		Response time	0.2937	0.2690	9.18
	Client	17.9976	18.5010	−2.72		Utilization	0.2968	0.2960	0.28
	N1	1.6660	1.6690	−0.18					
	N2	1.6660	1.6700	−0.24	RB-4-4	Place	Model	Sim	Error %
	N3	1.6660	1.6700	−0.24		Client	21.4974	22.1040	−2.74
						N1	1.2853	1.3380	−3.94
	Throughput	8.9988	9.2510	−2.73		N2	1.2853	1.3380	−3.94
	Response time	0.5554	0.4860	14.28		N3	1.2853	1.3370	−3.87
	Utilization	0.6249	0.6410	−2.51		N4	1.2853	1.3360	−3.79
						Throughput	10.7487	11.053	−2.75
						Response time	0.4783	0.4430	7.97
						Utilization	0.5624	0.5820	−3.36

of each place. Errors for overall mean response time are less than 20%. We note that choosing the client think rate as $\mu_T = 0.5$ allowed the mean number of customers in the net to be greater than the number of places in the RB-n-m.

Further, for the same service rates and replication factor, as the number of nodes increases the probability of a customer arrival at an input transition decreases, thus decreasing the reversed rates and accordingly the utilization of each node. This is evident from Table 2, when comparing the performance measures of the RB-2-2, RB-3-2 and RB-4-2 replication blocks. In Section 6.3, we show how to overcome this modelling limitation by composing smaller replication blocks.

5.2. Accuracy

To investigate the accuracy of the conditions stated in Conjecture 2, we evaluated each condition to identify cases that provided counter examples. This was conducted by comparing different configurations of RB-n-m models with simulations of the corresponding n-node fork-join queueing systems. We summarize our findings below.

$\frac{x_i}{\mu_i} < \frac{1}{n_i}$
This condition minimizes the effect of the non-synchronizing transitions on the behavior of the RB-n-m. Removal of this

Table 3
Counter examples for the accuracy conditions for an RB-n-m.

		Parameters				Relative error % compared to simulation				
		μ_i	μ_j	μ_T	# In system	Client	N_i	Throughput	Response time	Utilization
$\mu_i < \mu_j$	RB-5-4	5	12	10	2	−16.74	−7.79	−16.71	12.41	−16.78
	RB-6-3	3	40	0.6	4	9.31	8.07	9.34	1.65	9.35
	RB-5-4	5	12	0.5	9	−2.82	−2.08	−2.81	5.41	2.83
$\mu_i > \mu_j$	RB-4-2	20	2	100	3	−1.67	3.40	−1.69	13.57	−1.48
	RB-5-2	12	5	10	4	−7.17	1.69	7.73	−17.86	−6.51
$\mu_i = \mu_j$	RB-5-2	5	5	3	5	−7.68	0.78	8.29	−18.85	−7.94
	RB-6-5	2	2	0.4	6	2.74	5.74	2.74	3.85	1.39
	RB-4-2	5	5	0.9	11	−6.55	−2.07	−6.58	17.83	−6.93

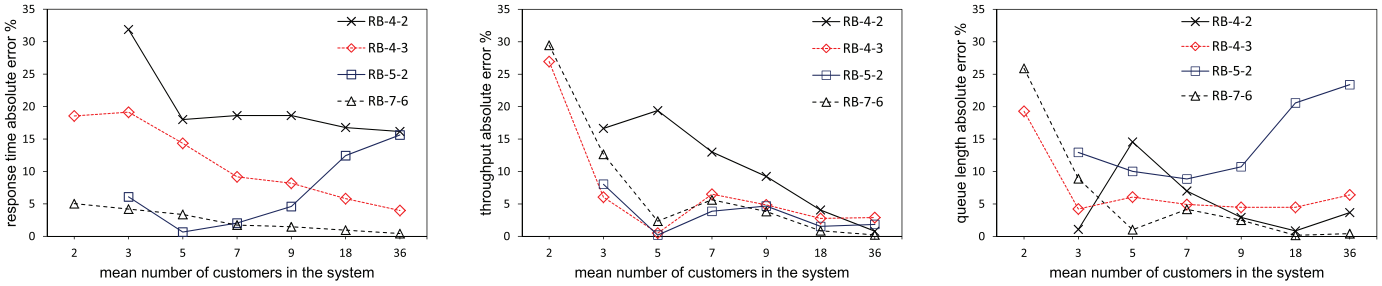

Fig. 5. The absolute error percentage vs. mean number of customers in the system for the (a) response time, (b) throughput and (c) mean queue length (N_i) for the RB-4-2, RB-4-3, RB-5-2 and RB-7-6 replication blocks.

constraint produces models that incorrectly estimate the mean response time, throughput and client queue length, even for small systems or large number of customers.

$\mu_i \neq \mu_j$ & $C > N$

Table 3 gives examples of RB-n-m replication blocks in which $\mu_i \neq \mu_j$ and $C \not> N$ that give accurate approximations compared to simulations of their corresponding fork-join systems. We note that the error percentages decrease as the number of customers increase, as evidenced by the second entry for the RB-5-4 with nine customers in the system in comparison to two customers in the system. Fig. 5 shows the absolute error percentage when comparing the RB-4-2, RB-4-3, RB-5-2 and RB-7-6 replication blocks with simulation for the (a) response time, (b) throughput and (c) mean queue length in each node versus the mean number of customers in the system. The absolute error percentages for the mean number of customers in the client place follow the same pattern as that of the throughput (not shown). Each replication block was configured with $\mu_i \neq \mu_j$ and μ_T was varied to produce the mean number of customers shown on the x-axis. We note that for the RB-4-2 and RB-7-6 the minimal number of customers produced by the model is three.

Most configurations observed have similar accuracy to that of the RB-4-3 and RB-7-6 in Fig. 5: there is a decrease in the absolute error percentage with the increase in the mean number of customers in the system. In addition, the accuracy of the solution is limited at small number of customers, e.g., some performance measures have absolute error rates above 20% when the mean number of customers in the system is less than three. For the RB-4-3 and RB-7-6, this is clear for the mean throughput and mean queue length in Fig. 5(b & c), while in contrast the absolute error for the mean response time is always below 20% irrespective of the mean number of customers. We note that the RB-4-2 replication block produces absolute errors percentages below 20% for all performance measures *only* when the number of customers is five and above, which adheres to the conditions of Conjecture 2.

For larger models, in some cases, product forms do not exist except when $\mu_i \gg \mu_j$ or $\mu_j \gg \mu_i$ and give similar accuracy to that observed above. However, when $n < 6$ and $\mu_i \gg \mu_j$ or $\mu_j \gg \mu_i$ the model becomes inaccurate when the mean number of customers is large. For example, in Fig. 5, the RB-5-2 was parameterized with $\mu_i = 3$ and $\mu_j = 20$. We notice that the absolute error is acceptable when the mean number of customers is less than 9. However, the absolute error for the response time and mean queue length increases as the mean number of customers increases above 9, with the absolute error for the mean queue length exceeding 20% at 18 and more customers. We observed similar behaviour for other models when $\mu_i \gg \mu_j$ or $\mu_j \gg \mu_i$ and $n < 6$ and in some cases when $\mu_i > \mu_j$.

$\mu_i = \mu_j$ & $C \gg N$

When $\mu_i = \mu_j$, most RB-n-m we investigated provided accurate approximations when $C < N$, however, the RB-4-2, shown in the last row of Table 3 did not provide accurate approximations compared to simulation until the number of customers was more than twice the number of places.

$C > N$ or $C >> N$

For any RB-n-m, with service rate μ_i, μ_j, μ_T, the solution of the rate equations will produce a fixed utilization for each place, that does not depend on μ_T, i.e., independent of the mean number of customers in the system. Accordingly, the mean number of customers, N_i, in each place will not depend on the mean number of customers in the system. If $N_i < 1$, the solution of the RB-n-m will be an inaccurate approximation of its corresponding n-node fork-join system. This can be rectified by increasing the mean number of customers in the system by increasing the think time of the client, which guarantees more customers in the system in the steady state. The conditions $C > N$ or $C >> N$ ensure that in equilibrium, after a long period, the effect of the non-synchronizing output transitions on the behavior of the RB-n-m diminish and tend to the characteristics of the n-node fork-join queueing system.

Furthermore, for any RB-n-m, as n grows larger, the routing probability to each input transition becomes smaller and,

Table 4
Parameters and error percentages for the RB-5-x replication block in comparison to the 5-node fork-join queueing system in which the replicated requests were equi-likely to the non-replicated requests.

RB-5-m	Model			Error % compared to simulation ($\sum p_i = \sum p_j$)			
	μ_T	$\sum p_i$	$\sum p_j$	Throughput	Response time	Utilization	N_i
2	0.36	0.51	0.49	0.69	−9.11	0.57	−4.52
3	0.135	0.91	0.09	0.36	−11.19	9.42	−13.60
4	0.18	0.98	0.02	1.17	2.61	24.95	−33.66
5	0.54	0.97	0.03	−7.22	31.98	28.32	−57.93

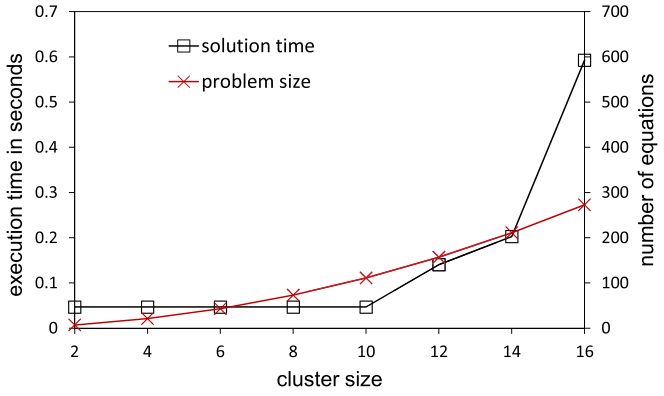

Fig. 6. Comparison between the solution evaluation time and the optimization problem size for different cluster sizes with $m = 2$.

correspondingly, the reversed rates. Hence, the utilization per place and N_i decrease as n increases (refer to Table 2 for examples). When $C < N$, for most configurations, we have found that the analytical model will provide a poor approximation for response times and throughput, while maintaining accuracy for mean queue length at each node. We note that as n increases, the increasing number of constraints can become difficult to satisfy in the case of a heterogeneous model.

5.3. Scalability

To evaluate the scalability of our solution method we compared the size of the optimization problem, in terms of the number of equations, with the solution time in seconds for replication blocks RB-n-2, where $n = 2, \ldots, 16$, i.e. with $m = 2$. The solution time includes the time to generate and solve the $n + 1 + \binom{n}{m}$ rate equations, the $\binom{n}{m}$ product-form conditions and the n accuracy conditions. Fig. 6 shows the problem size increasing linearly with the increase in cluster size. The solution time is constant – less than 0.05 s – up to replication block RB-10-2, which uses 111 equations, and increases to 0.6 s for the RB-16-2 replication block which has an optimization problem with 273 equations.

In addition, we compared our solution method with current RCAT implementations, mainly AutoCAT (Casale and Harrison, 2013) and INAP (Balsamo et al., 2010a; Marin and Bulo, 2009). AutoCAT (Casale and Harrison, 2013) formulates the RCAT sufficient conditions for product-form into a non-linear optimization problem with non-convex quadratic constraints. INAP (Balsamo et al., 2010a; Marin and Bulo, 2009) is a fixed point iterative technique to compute the stationary distributions of product-form models that satisfy MARCAT (Harrison and Lee, 2005). Both methods are limited to models with pairwise synchronizations and thus are only suitable for the RB-n-m models when $m = 2$. Comparing against an open model comprising a BB-2 and three queues (equivalent to 13 equations), our method provided a more efficient solution time of 0.09 s, with AutoCAT and INAP taking 35 and 234 seconds respectively (Casale and Harrison, 2013).

6. Modelling replication in NoSQL datastores

NoSQL cloud datastores have been designed for horizontal scalability and high availability through the replication of data across different machines and different data centers. For a cluster of n nodes, the replication factor is the number of times a data item is duplicated across the cluster. Clients request two classes of data: *non-replicated* requests, which is data that requires weak consistency guarantees and *replicated* requests, which is data that requires high consistency guarantees. In distributed NoSQL single-master datastores (Cattell, 2011; Stonebraker and Cattell, 2011), e.g., Redis (0000), the client application is aware of the location of the replicated data within the datastore cluster. For any request, the client will contact the designated node that holds a copy of the data item. If the client requires a higher consistency guarantee, it will contact more than one node in parallel and resolve the returned values internally.

Assume a single-master datastore with n nodes and replication factor m with replicated and non-replicated requests arriving randomly from clients. We represent the clients and the n-node cluster with m replication factor as an RB-n-m replication block. In the following we study the effect of replication and cluster size on the performance of single-master datastores and illustrate the composition of larger clusters from smaller RB-n-m blocks.

6.1. Effect of replication on performance

Request response time and throughput in a single-master datastore depends on the number of nodes and the level of replication of data across the cluster. We investigate the effect of replication on performance of such systems using a 5-node single-master datastore modeled as an RB-5-m. We compare the mean response times and mean throughput for the 5-node cluster for different replication factors: $m = 2, \ldots, 5$. The service rate for the non-replicating output transitions was $\mu_i = 5$ and for the replicating output transitions was $\mu_j = 12$. The client think rate was chosen so that the mean number of customers in the system was always 30. Table 4 gives the values of μ_T for each replication factor. When comparing the performance measure of the RB-5-m to that of a simulation of the corresponding 5-node fork-join queueing system (not shown), as expected, errors were less than 10%.

From Fig. 7, as the replication factor increases the mean response time and throughput decrease, with a noticeable increase when $m = 5$, i.e., the full fork case. The decrease in throughput from $m = 2$ to $m = 3$ results from the decrease in utilization (mean queue length) of the queues due to the distribution of requests across the cluster with the increase in replication. This translates into a corresponding decrease in response time. From $m = 3$ to $m = 4$, we observe an increase in throughput and response time, as the mean queue length increases along with the response time and utilization. However, this increase is less than the performance measures for the case when $m = 2$. For $m = 5$, i.e., full fork, the mean queue length is at its maximum giving maximum utilization and throughput. This example illustrates the trade-off between consistency (higher replication) and

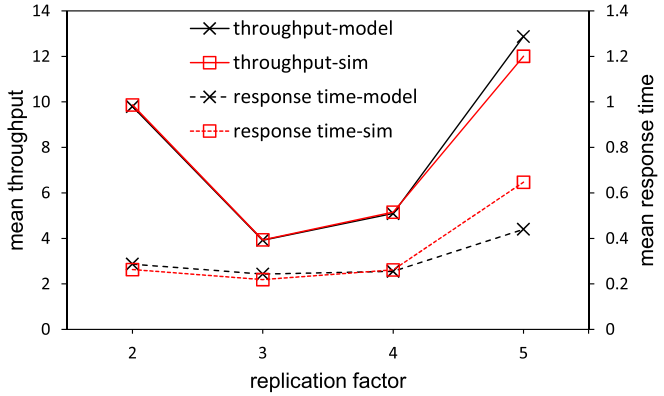

Fig. 7. The mean throughput and mean response time for the RB-5-m with $\mu_i = 5$ and $\mu_j = 12$ for different replication factors in comparison to a simulated RB-5-m where $\sum p_i = \sum p_j = 0.5$.

performance (lower utilization/response times) that needs to be considered when configuring cloud datastores.

To attain a product-form solution for any RB-n-m, the values of the arrival rates (reversed rates) of the input transitions are constrained only to values that satisfy the conditions in Section 3. These values are determined by the client nodes routing probabilities. As seen from Table 1 and discussed in Section 5.1, the product-form solution differentiates between replicated and non-replicated incoming requests such that, in most cases, the probability of the client generating a replicated and non-replicated request is never equal, i.e., $\sum p_i \neq \sum p_j$. In fact, for most cases, no product-form exists for the case when $\sum p_i = \sum p_j$. If the system to be modeled presents a similar distribution of requests as that of the solution of the model, accuracy is guaranteed. However, in realistic datastore workloads that is not the case. To investigate the applicability of the RB-n-m to real-world scenarios, we compared the RB-5-m to a simulation of a 5-node fork-join system, with $\mu_i = 5$ and $\mu_j = 12$, in which $\sum p_i = \sum p_j = 0.5$. This is referred to as *sim* in Fig. 7.

Fig. 7 compares the mean response time and the mean throughput of the RB-5-m to the simulated system in which $\sum p_i = \sum p_j = 0.5$. We can see that the model follows the trend of the simulated system for all replication factors. Table 4 shows the RB-5-m parameters and the total probabilities for the replicated and non-replicated requests. From Fig. 7 and Table 4 the RB-5-m ($m = 2, 3, 4$) accurately approximates the mean throughput and response times for the corresponding system, even though the distribution of requests for the RB-5-3 and RB-5-4 are heavily skewed towards the non-replicated requests. However, the RB-5-4 inaccurately approximates the utilization of the node as a result of inaccurately approximating the mean queue length. The RB-5-5 only approximates the mean throughput correctly. The skew towards non-replicated requests in the RB-5-5 directly affects the models ability to correctly approximate the performance measures of the simulated system.

In general, when comparing to actual systems, the RB-n-m models can be used to give an indication of the trend of performance measures for different replication factors in n-node clusters. Accuracy of the approximations are excellent when the distribution of requests are equal or reasonably similar to that of the actual system. In Section 6.3, we show how to compose models to achieve a more realistic distribution of requests.

6.2. Effect of cluster size on performance

To investigate the effect of the cluster size on the performance of a single-master datastore, we compare the performance of the RB-n-m, where $n = 3, \ldots, 7$ and $m = 3$ for clusters of size four and above and $m = 2$ when the cluster size is three. The models were parameterized with $\mu_i = \mu_j = 1$ as this was the only configuration that provided a product-form solution for all the replication blocks. The client think rate was chosen so that the mean number of customers in the system was always 30. The solution was compared to a simulation of the corresponding n-node fork-join system (not shown) producing errors of less than 18%.

Fig. 8(a) shows the mean response time and mean throughput and (b) the mean utilization and mean queue length for the different cluster sizes with fixed number of customers. Intuitively, the increase in cluster size produces a decrease in all performance measures. As the cluster size increases, the same workload is distributed over more nodes, thus decreasing the mean queue length and utilization at each node which results in a decrease in mean throughput and response time. We compared the RB-n-m solution to that of a simulated n-node fork-join system, with $\mu_i = \mu_j = 1$, in which $\sum p_i = \sum p_j = 0.5$ as in the previous section. From Fig. 8(a) & (b), the model follows the trend of the simulation and produces similar values for the mean throughput.

6.3. Composition of clusters

The RB-n-m models presented in the previous sections assumed that the data items are distributed evenly across the cluster. In more realistic scenarios, data item placement is based on user access behavior, data center proximity and storage availability. Furthermore, in practice, industry deployments of NoSQL datastores favor partial consistency in which the number of copies of data items replicated within the cluster is three or below (Bailis et al., 2012). This means

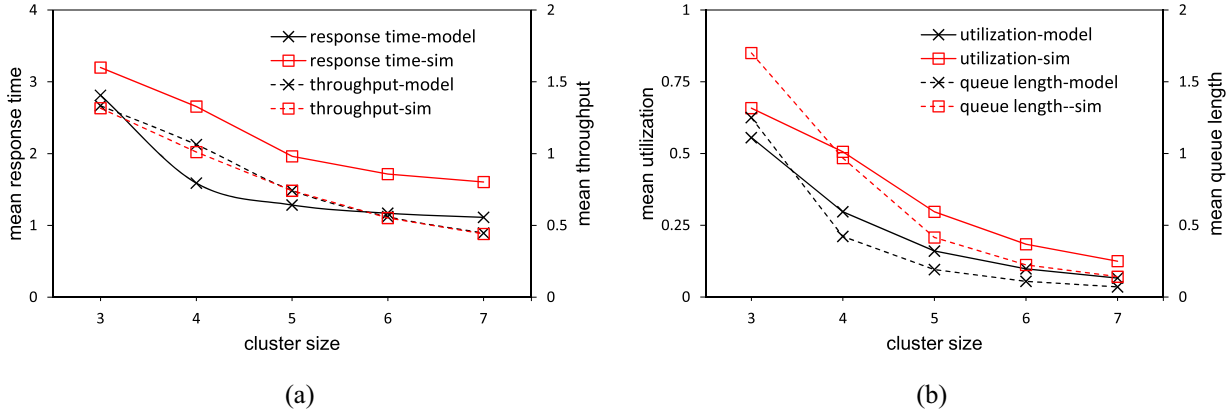

Fig. 8. The (a) mean response time and throughput and (b) mean utilization and queue length for replication blocks of sizes 3–7 in comparison to simulated clusters where $\sum p_i = \sum p_j = 0.5$.

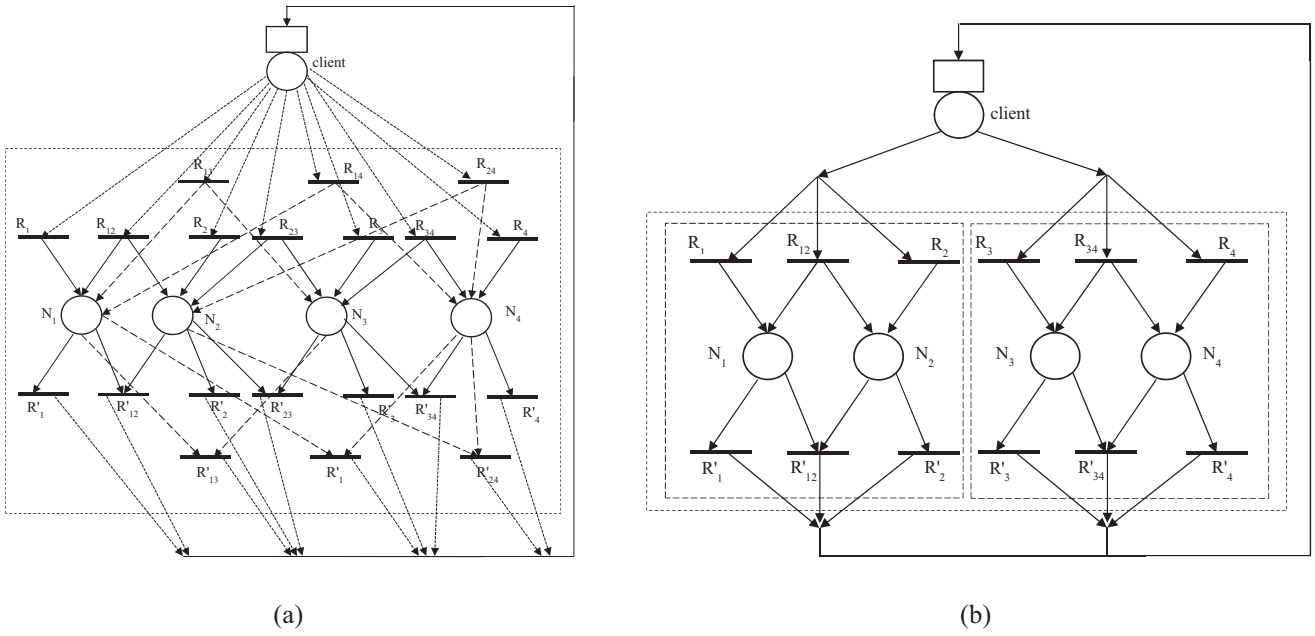

Fig. 9. The (a) RB-4-2 and (b) the composed 4-node cluster from two RB-2-2 clusters.

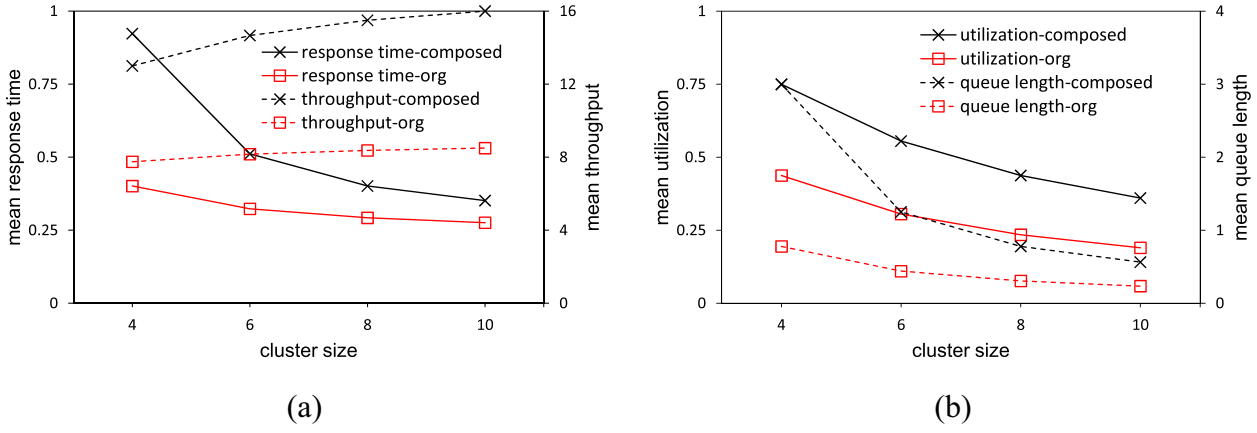

Fig. 10. The (a) mean response time and throughput and (b) mean utilization and queue length for composed RB-n/2-m clusters of sizes 4–10 with replication factor $m = 2$ in comparison to RB-n-m clusters.

that a cluster of n nodes with replication factor 3, will designate only 3 nodes to hold a certain data item. The structure of most NoSQL datastores is symmetric, thus we can model a large n node cluster with replication factor m as a set of smaller RB-x_i-m clusters in which $n = \sum x_i$. For example, Fig. 9(a) shows an RB-4-2 cluster with random access across the cluster. Fig. 9(b) shows a 4 node cluster in which the data items are replicated twice on two fixed nodes. This is represented by two RB-2-2 clusters.

The solution of the model in Fig. 9(b) follows the same technique as that stated in Section 5 with the modification of the rate equations to represent the distribution of the incoming requests between the two smaller clusters. A product-form solution is possible as the RCAT theorem allows for the composition of BB-n and product-form constructs to form larger models (Lladó and Harrison, 2011; Balsamo et al., 2012). A further advantage of composing RB-n-m models is that the smaller RB-n-m models are more flexible when finding product-form solutions, thus expanding the scope of scenarios that can be modeled.

In Fig. 10, we model n-node clusters where $n = 4, \ldots, 10$ as a set of two RB-n/2-m clusters and compare their performance to the corresponding RB-n-m clusters. The models were parameterized with $\mu_i = 4$, $\mu_j = 12$ and the client think times where chosen such that the mean number of customers in the system are 100. The number of copies in the cluster was fixed at $m = 2$ for both the large and smaller clusters. The solution of the models was compared to a simulation of the corresponding n-node fork-join system (not shown) producing errors of less than 6%. From Fig. 10(a & b), we notice that both configurations exhibit the same performance trends. As the cluster size increases the mean response time, mean queue length and mean utilization decreases while the mean throughput increases. However, the magnitude of the performance measures for the RB-n/2-m clusters is larger than the RB-n-m clusters, as the percentage of traffic passing through each individual cluster is higher than that of the corresponding larger cluster. In particular, the RB-n/2-m clusters give higher utilizations and throughput than the corresponding RB-n-m clusters, especially for larger cluster sizes, which is more representative for capacity planning.

To further utilize the flexibility of smaller clusters, a large RB-n-m can be composed of any combination of smaller clusters composed in a way that utilizes the symmetry and inherent load balancing of NoSQL datastore architectures. Furthermore, smaller clusters allow for realistic routing probabilities to the input transitions within the

clusters and the larger models can be solved with different percentages of traffic entering each small cluster giving rise to different modelling scenarios. In addition, RB-n-m models can be composed with other product-form constructs to form detailed models of network infrastructure, cloud storage and data center delays, as illustrated in (Lladó and Harrison, 2011).

7. Conclusions

This paper presented a product-form approximation of closed fork-join queueing systems with interfering requests using SPNs in the form of RB-n-m replication blocks under conditions for the SPN transitions. This has been illustrated through the modelling of single-master replication in NoSQL datastores in which the RB-n-m gave excellent accuracy. In addition, we have demonstrated the ability to compose smaller RB-n-m replication blocks to form larger clusters with more realistic properties. The scalability and resource efficiency of the solution of the RB-n-m models makes them amenable to lightweight incorporation within autonomous monitoring and feedback systems for cloud datastores.

We have shown that it is possible to accurately incorporate replication and synchronization within larger Markovian models without undue costs of analysis. Our method of directly solving the RCAT rate equations as a nonlinear optimization problem eliminates the need for representing the underlying CTMC of the cooperating Markovian processes and thus allowing for the modelling and analysis of large and complex systems. In addition, the ability to integrate user-defined constraints on the variables and parameters in the optimization problem provides the modeller with the flexibility to experiment with different configurations and parameterizations, thus imposing pre-defined conditions on the feasible solution, and producing different product-form solutions when they exist. Further, as the solution method described in this paper is not specific to product-form RB-n-m replication blocks it can be efficiently applied to solve large models with heterogeneous cooperating agents that satisfy the RCAT conditions.

Our models have focused on the RB-*n*-*m* class of replication blocks, in which non-replicated requests are sent to each of the n nodes in a cluster and all combinations of m nodes – $\binom{n}{m}$ of them – also receive replicated requests, i.e. full and partial fork-join systems. It is straightforward to model replication in which not every combination of the m nodes is available for replication and also more or less than m nodes may be used; giving an RB-n-$\{m_1, \ldots, m_k\}$ building block, where $1 \le m_j \le n$, $j = 1, \ldots, k$, in the event that all combinations of $\{m_1, \ldots, m_k\}$ nodes are available, for example. The method applied in this paper is applicable to open systems, and as the product-form constraints are less restrictive, more representative scenarios can be explored. We would need to investigate the accuracy conditions of such product-form models when approximating fork-join queueing systems with interfering requests and configurable forks.

Appendix. Product-forms using RCAT

Assume we have a set of cooperating Markovian processes: $M = \{P_1, \ldots, P_N\}$. A state transition in the CTMC of the network is denoted by an *action*. A *cooperation* between two *components* or processes involves a set of actions A, with labels a. Each action a represents a synchronizing transition in a pairwise cooperation between processes P_i and P_j $(i \neq j)$, which takes place in both processes simultaneously. If an action a causes a state transition $(s_i \to s_{i'})$ in P_i and a corresponding transition from $(s_j \to s_{j'})$ in P_j $(i \neq j)$, both with rate μ_a, then a is considered an *active* action in P_i and a *passive* action in P_j. For simplicity we will use a simple example to illustrate the *application* of RCAT.

To derive a product-form using RCAT the *reversed rates* of the synchronizing active actions must be calculated. The reversed rate for a transition from state i to state j with rate λ is (Kelly, 1979):

$$\overline{r_\lambda} = \frac{\pi(i)\lambda}{\pi(j)} \tag{16}$$

where $\pi(\cdot)$ denotes the equilibrium probability function of the CTMC.

Assume we have an open queueing network with three M/M/1 nodes in tandem: P_1, P_2, P_3, each with service rates μ_1, μ_2, μ_3, respectively. External Poisson arrivals arrive at P_1 with rate λ_1, are serviced and immediately proceed to P_2, then to P_3 and leave the network after processing. Therefore, $M = \{P_1, P_2, P_3\}$ and $A = \{\lambda_1, \mu_1, \mu_2, \mu_3\}$. Let R_i denote the *isolated* component for the process P_i. The goal of RCAT is to find the rates of these isolated components in order to calculate the product-form. The steps to apply RCAT are as follows (Harrison, 2010):

1. For each P_i, construct R_i by setting the rate of every instance $a \in A$ that is passive in P_i to x_a. Hence:

   $R_1 \sim \{\lambda_1, \mu_1\}$ (no passive actions),
   $R_2 \sim \{x_{\mu_1}, \mu_2\}$ (μ_1 is a passive action in P_2),
   $R_3 \sim \{x_{\mu_2}, \mu_3\}$ (μ_2 is a passive action in P_3).

   where we have used the symbol $\sim$ to indicate that a process is given by its defining instantaneous transition rates.
2. For each active action $a \in R_i$, check that the reversed rate, $\overline{r_a}$ is the same for all its instances. Applying (16) we have:

   $\overline{r_{\mu_1}} = \lambda_1$,
   $\overline{r_{\mu_2}} = x_{\mu_1}$,
   $\overline{r_{\mu_3}} = x_{\mu_2}$.

   We note that the reversed rate $\overline{r_a}$ will in general be a function of x_b, where $b \in A$.
3. Solve the *rate equations*: $x_a = \overline{r_a}$ for each $a \in A$. This step gives:

   $x_{\mu_1} = \overline{r_{\mu_1}} = \lambda_1$,
   $x_{\mu_2} = \overline{r_{\mu_2}} = x_{\mu_1}$.

   which simplifies to:

   $x_{\mu_1} = x_{\mu_2} = \lambda_1$.
4. Substituting the solution of the rate equations in R_i gives:

   $R_1 \sim \{\lambda_1, \mu_1\}$,
   $R_2 \sim \{\lambda_1, \mu_2\}$,
   $R_3 \sim \{\lambda_1, \mu_3\}$.
5. Check that the RCAT enabling conditions (Harrison, 2003) stated below are satisfied.
6. The required product-form for state $\mathbf{s} = (s_1, s_2, s_3)$ is $\pi(\mathbf{s}) \propto \pi_1(s_1)\pi_2(s_2)\pi_3(s_3)$ where $\pi_k(s_k)$ is the equilibrium probability of state s_k in R_k. In this case:

$$\pi(s_1, s_2, s_3) = \prod_{i=1,2,3} (1 - \rho_i)\rho_i^{s_i},\ \rho_i = \lambda_1/\mu_i.$$

Informally, the RCAT conditions require (Harrison, 2003; Harrison and Marin, 2013):

1. If a synchronizing type t is passive in a component, then all states of that component must have one outgoing transition with type t;
2. If a synchronizing type t is active in a component, then all states of that component must have one incoming transition with type t;
3. In the isolated components, all transitions sharing the same active synchronizing type must have the same reversed rate.

For queueing networks, all passive actions are enabled in all states and all states have an incoming instance of every active action. This is straight forward for an $M/M/1$. For the formal definition and proof of RCAT and its extension to two way and multi-way synchronizations the reader is referred to (Harrison, 2003, 2004; Harrison and Lee, 2005).

The elegance of RCAT and its extensions (Harrison, 2004; Harrison and Lee, 2005), has been in providing a unified automated approach to deriving product-form solutions to a variety of known cooperating Markovian processes: i.e., Jackson queueing networks (Harrison, 2003), BCMP networks (Harrison, 2004), G-networks with triggers (Harrison and Marin, 2013) and stochastic Petri nets (Balsamo et al., 2012). Moreover, Extended RCAT has produced new product-form solutions for G-networks with generalized resets (Harrison, 2004), finite capacity queues with blocking or skipping (Balsamo et al., 2010b) and stochastic Petri nets with signals (Marin et al., 2012).

Current work in automating the application of RCAT and the identification and/or solution of the rate equations falls into two categories:

- Algorithms to derive the rate equations for specific Markovian processes, e.g., in Harrison and Marin (2013) the authors describe a general algorithm to derive the rate equations for any set of cooperating Markovian processes specified in PEPA (Hillston, 1994). In Balsamo et al. (2012), an algorithm to identify the basic *building blocks* of a stochastic Petri net is described. Even though these algorithms have not been automated, the implementation is straightforward and can be generalized to identifying reversed rates and the RCAT rate equations from XML representations of Markovian models.
- Automated algorithms to compute the values of the reversed rates by applying RCAT conditions to the CTMCs of the cooperating agents (Casale and Harrison, 2013; Balsamo et al., 2010a; Marin and Bulo, 2009). This is conducted without direct use or solution of the rate equations. These methods solve subsets of RCAT models and are computationally expensive in comparison to the method described in this paper. Moreover, the description of the input is not straightforward, as in some cases the CTMC must be truncated in order for a feasible solution to be calculated.

References

Alomari, F., Menasce, D., 2014. Efficient response time approximations for multiclass fork and join queues in open and closed queuing networks. IEEE Trans. Parallel Distrib. Syst. 25 (6), 1437–1446. doi:10.1109/TPDS.2013.70.

Baccelli, F., 1985. Two Parallel Queues Created by Arrivals With Two Demands: The M/G/2 Symmetrical Case. Technical Report INRIA No. 426. INRIA, Paris, France.

Baccelli, F., Makowski, A., 1985. Simple computable bounds for the fork-join queue 436–441.

Baccelli, F., Makowski, A., 1989. Queueing models for systems with synchronization constraints. Proceedings of the IEEE 77 (1), 138–161. doi:10.1109/5.21076.

Bailis, P., Venkataraman, S., Franklin, M.J., Hellerstein, J.M., Stoica, I., 2012. Probabilistically bounded staleness for practical partial quorums. Proc. VLDB Endow. 5 (8), 776–787. doi:10.14778/2212351.2212359.

Balsamo, S., Dei Rossi, G.-L., Marin, A., 2010a. A numerical algorithm for the solution of product-form models with infinite state spaces. In: Proceedings of the EPEW 2010. In: *LNCS*, 6342, pp. 191–206. doi:10.1007/978-3-642-15784-4_13.

Balsamo, S., Donatiello, L., Van Dijk, N.M., 1998. Bound performance models of heterogeneous parallel processing systems. IEEE Trans. Parallel Distrib. Syst. 9 (10), 1041–1056. doi:10.1109/71.730531.

Balsamo, S., Harrison, P.G., Marin, A., 2010b. A unifying approach to product-forms in networks with finite capacity constraints. In: Proceedings of the International Conference on Measurement and Modeling of Computer Systems, pp. 25–36. doi:10.1145/1811039.1811043.

Balsamo, S., Harrison, P.G., Marin, A., 2012. Methodological construction of product-form stochastic petri nets for performance evaluation. Journal of Systems and Software 85 (7), 1520–1539. http://dx.doi.org/10.1016/j.jss.2011.11.1042.

Balsamo, S., Mura, I., 1997. On queue length moments in fork and join queuing networks with general service times. In: Marie, R., Plateau, B., Calzarossa, M., Rubino, G. (Eds.), Computer Performance Evaluation Modelling Techniques and Tools. In: *Lecture Notes in Computer Science*, 1245. Springer Berlin Heidelberg, pp. 218–231. doi:10.1007/BFb0022209.

Casale, G., Harrison, P.G., 2013. Autocat: automated product-form solution of stochastic models. In: Matrix-Analytic Methods in Stochastic Models. In: *Springer Proceedings in Mathematics & Statistics*, 27, pp. 57–85. doi:10.1007/978-1-4614-4909-6_4.

Casale, G., Muntz, R., Serazzi, G., 2008. Geometric bounds: a noniterative analysis technique for closed queueing networks. IEEE Trans. Comput. 57 (6), 780–794. doi:10.1109/TC.2008.37.

Cattell, R., 2011. Scalable sql and nosql data stores. SIGMOD Rec. 39 (4), 12–27. doi:10.1145/1978915.1978919.

Chen, R., 2001. A hybrid solution of fork/join synchronization in parallel queues. IEEE Trans. Parallel Distrib. Syst. 12 (8), 829–845. doi:10.1109/71.946659.

Coulden, D., Osman, R., Knottenbelt, W.J., 2013. Performance modelling of database contention using queueing petri nets. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, pp. 331–334.

Dean, J., Ghemawat, S., 2008. Mapreduce: Simplified data processing on large clusters. Commun. ACM 51 (1), 107–113. doi:10.1145/1327452.1327492.

Di Sanzo, P., Palmieri, R., Ciciani, B., Quaglia, F., Romano, P., 2010. Analytical modeling of lock-based concurrency control with arbitrary transaction data access patterns. In: Proceedings of the WOSP/SIPEW '10, pp. 69–78. doi:10.1145/1712605.1712619.

Elnikety, S., Dropsho, S., Cecchet, E., Zwaenepoel, W., 2009. Predicting replicated database scalability from standalone database profiling. In: Proceedings of the EuroSys '09, pp. 303–316. doi:10.1145/1519065.1519098.

Flatto, L., Hahn, S., 1984. Two parallel queues created by arrivals with two demands. I. SIAM J. Appl. Math. 44 (5), 1041–1053.

Gandini, A., Gribaudo, M., Knottenbelt, W.J., Osman, R., Piazzolla, P., 2014. Performance analysis of nosql databases. In: Proceedings of the 11th European Performance Engineering Workshop (EPEW 2014).

Harrison, P., Zertal, S., 2003. Queueing models with maxima of service times. In: Kemper, P., Sanders, W. (Eds.), Computer Performance Evaluation. Modelling Techniques and Tools. In: *Lecture Notes in Computer Science*, 2794. Springer Berlin Heidelberg, pp. 152–168. doi:10.1007/978-3-540-45232-4_10.

Harrison, P.G., 2003. Turning back time in Markovian process algebra. Theor. Comput. Sci. 290 (3), 1947–1986.

Harrison, P.G., 2004. Reversed processes, product forms and a non-product form. Linear Algebra Appl. 386 (0), 359–381.Special Issue on the Conference on the Numerical Solution of Markov Chains 2003. http://dx.doi.org/10.1016/j.laa.2004.02.020

Harrison, P.G., 2010. Turning back time - what impact on performance? Comput. J. 53 (6), 860–868.

Harrison, P.G., Lee, T.T., 2005. Separable equilibrium state probabilities via time reversal in markovian process algebra. Theor. Comput. Sci. 346 (1), 161–182.

Harrison, P.G., Marin, A., 2013. Product-forms in multi-way synchronizations. Comput. J. doi:10.1093/comjnl/bxt103.

Hillston, J., 1994. A Compositional Approach to Performance Modelling. University of Edinburgh Ph.D. thesis.

Huebscher, M.C., McCann, J.A., 2008. A survey of autonomic computing—degrees, models, and applications. ACM Comput. Surv. 40 (3), 7:1–7:28. doi:10.1145/1380584.1380585.

Kelly, F., 1979. Reversibility and Stochastic Networks. Cambridge University Press.

Kumar, A., Shorey, R., 1993. Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system. IEEE Trans. Parallel Distrib. Syst. 4 (10), 1147–1164. doi:10.1109/71.246075.

Lebrecht, A., Knottenbelt, W., 2007. Response time approximations in fork-join queues, pp. 12–19.

Lladó, C.M., Harrison, P.G., 2011. A pmif with petri net building blocks. In: Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering, pp. 103–108. doi:10.1145/1958746.1958764.

Marin, A., Balsamo, S., Harrison, P.G., 2012. Analysis of stochastic petri nets with signals. Perform. Eval. 69 (11), 551–572. doi:10.1016/j.peva.2012.06.003.

Marin, A., Bulo, S., 2009. A general algorithm to compute the steady-state solution of product-form cooperating markov chains. In: Proceedings of the IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, MASCOTS '09., pp. 1–10. doi:10.1109/MASCOT.2009.5366744.

Menasce, D., 2004. Composing web services: A qos view. Internet Comput., IEEE 8 (6), 88–90. doi:10.1109/MIC.2004.57.

Nelson, R., Tantawi, A., 1988. Approximate analysis of fork/join synchronization in parallel queues. IEEE Trans. Comput. 37 (6), 739–743. doi:10.1109/12.2213.

Nicola, M., Jarke, M., 2000. Performance modeling of distributed and replicated databases. IEEE Trans. Knowl. Data Eng. 12 (4), 645–672. doi:10.1109/69.868912.

Osman, R., Awan, I., Woodward, M.E., 2011. Queped: Revisiting queueing networks for the performance evaluation of database designs. Simul. Model. Pract. Theory 19 (1), 251–270.

Osman, R., Coulden, D., Knottenbelt, W.J., 2013. Performance modelling of concurrency control schemes for relational databases. In: Proceedings of the ASMTA, pp. 337–351.

Osman, R., Knottenbelt, W.J., 2012. Database system performance evaluation models: a survey. Perform. Eval. 69 (10), 471–493.

Osman, R., Piazzolla, P., 2014. Modelling replication in nosql datastores. In: Proceedings of the 11th International Conference on Quantitative Evaluation of Systems (QEST).

Patterson, D.A., Gibson, G., Katz, R.H., 1988. A case for redundant arrays of inexpensive disks (raid). SIGMOD Rec. 17 (3), 109–116. doi:10.1145/971701.50214.

Redis, . http://redis.io/.

Stonebraker, M., Cattell, R., 2011. 10 rules for scalable performance in 'simple operation' datastores. Commun. ACM 54 (6), 72–80. doi:10.1145/1953122.1953144.

Thomasian, A., 1997. Approximate analysis for fork/join synchronization in raid5. Comput. Syst.: Sci. Eng. 12 (5), 329–337.

Thomasian, A., Menon, J., 1994. Performance analysis of raid5 disk arrays with a vacationing server model for rebuild mode operation. In: Proceedings of the Tenth International Conference on Data Engineering, February 14-18, 1994, Houston, Texas, USA. IEEE Computer Society, pp. 111–119.

Thomasian, A., Menon, J., 1997. Raid5 performance with distributed sparing. IEEE Trans. Parallel Distrib. Syst. 8 (6), 640–657. doi:10.1109/71.595583.
Varki, E., 1999. Mean value technique for closed fork-join networks. SIGMETRICS Perform. Eval. Rev. 27 (1), 103–112. doi:10.1145/301464.301484.
Varki, E., 2001. Response time analysin of parallel computer and storage systems. IEEE Trans. Parallel Distrib. Syst. 12 (11), 1146–1161. doi:10.1109/71.969125.
Varki, E., Dowdy, L., Zhang, C., 2013. Quick performance bounding techniques for computer and storage systems with parallel resources. unpublished. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.4105.
Varki, E., Merchant, A., Chen, H., 2012. The m/m/1 fork-join queue with variable subtasks. unpublished. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.3062.
Varma, S., Makowski, A.M., 1994. Interpolation approximations for symmetric fork-join queues. Perform. Eval. 20 (1-3), 245–265. doi:10.1016/0166-5316(94)90016-7.

Rasha Osman is a post-doctoral research associate at the Department of Computing, Imperial College London, which she joined in 2011. She obtained a B.Sc. in Computer Science with Honors in 1995 and an M.Sc. in Computer Science in 2001 both from the Faculty of Mathematical Sciences, University of Khartoum, Sudan. After her studies she worked as a lecturer and a software developer in Sudan for 10 years. She completed her Ph.D. in Software Performance Engineering in 2010 at the University of Bradford, UK. Her main interests are in performance modeling and evaluation of database/datastore systems for real-time decision making, specifically in autonomous DBMSs and cloud datastores. She is a Fellow of the Higher Education Academy (UK), Senior Member of the IEEE & Member of the ACM.

Peter Harrison is Professor of Mathematical Modelling in the Department of Computing at Imperial College London, where he became a lecturer in 1983. He graduated at Christ's College Cambridge as a Wrangler in Mathematics in 1972 and went on to gain Distinction in Part III of the Mathematical Tripos in 1973, winning the Mayhew prize for Applied Mathematics. He obtained his Ph.D. in Computing Science at Imperial College in 1979. He has researched into stochastic performance modelling and algebraic program transformation for some thirty five years, visiting IBM Research Centers during two summers. He has written two books, had over 200 research papers published and held a series of research grants, both national and international. The results of his research have been exploited extensively in industry, forming an integral part of commercial products such as Metron's Athene Client–Server capacity planning tool. Currently, his main research interests are in stochastic modelling, where he has developed the RCAT methodology for finding separable solutions, Hidden Markov Models, response time analysis and modulated fluid models, together with applications such as storage systems, resource virtualization and energy-saving. He has taught a range of subjects at undergraduate and graduate level, including Operating Systems: Theory and Practice, Functional Programming, Parallel Algorithms and Performance Analysis.